Learning Truthful Mechanisms without Discretization

Yunxuan Ma
CFCS, School of Computer Science
Peking University
Beijing, China
yunxuanma@pku.edu.cn

Steven (Siqiang) Wang
Foster School of Business
University of Washington
Seattle, USA
siqiaw2@uw.edu

Zhijian Duan
CFCS, School of Computer Science
Peking University
Beijing, China
zjduan@pku.edu.cn

Yukun Cheng*
School of Business
Jiangnan University
Wuxi, China
ykcheng@amss.ac.cn

Xiaotie Deng*
CFCS, School of Computer Science
& CMAR, Institute for AI
Peking University
Beijing, China
xiaotie@pku.edu.cn

ABSTRACT

This paper introduces TEDI (Truthful, Expressive, and Dimension-Insensitive approach), the first discretization-free algorithm to learn truthful mechanisms. Existing learning-based algorithms rely on discretization of outcome spaces to ensure truthfulness, which suffers from inefficiency as problem size increases. To address this limitation, we formalize the concept of *pricing rules*, defined as functions that map outcomes to prices. We then parameterize pricing rules using Partial GroupMax Network, a novel network architecture designed to universally approximate partial convex functions. To enable optimization, we develop two training techniques: *covariance trick* and *continuous sampling*, to derive unbiased gradient estimators compatible with first-order optimization. Together, these design choices ensure the truthfulness, expressiveness and dimension-insensitivity of TEDI, and our experiments show that it consistently outperforms state-of-the-art methods in medium-to-large scale problems. [1]

KEYWORDS

Automated Mechanism Design; Differentiable Economics; Deep Learning; First-Order Algorithm

ACM Reference Format:
Yunxuan Ma, Steven (Siqiang) Wang, Zhijian Duan, Yukun Cheng*, and Xiaotie Deng*. 2026. Learning Truthful Mechanisms without Discretization. In *Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), Paphos, Cyprus, May 25 – 29, 2026*, IFAAMAS, 9 pages. https://doi.org/10.65109/JTXL5273

1 INTRODUCTION

Mechanism design is a crucial topic with broad applications in multi-agent systems and microeconomics. It examines how one mechanism designer can incentivize participants truthfully reporting their private information, while achieving better resources allocations. Mechanism design problems have been widely studied in variable contexts, including multiple goods monopolists [11, 21, 34, 39, 52], digital goods auctions [8, 22, 23], mechanism design for LLMs [16, 45, 46] and many other applications.

Meanwhile, differentiable economics—a technical paradigm developed recently [10, 19, 32, 49]—employs differentiable functions (e.g., neural networks) and gradient-based optimization to represent and to learn economic solutions. Within this framework, notable progress has been made in various mechanism design problems, including unit-supply auctions [9, 19, 47, 50], facility location [24], social choice [38], and stable matching [41].

However, all existing approaches guaranteeing truthfulness of learned mechanisms (e.g., MenuNet [44], Lottery-AMA [9], GemNet [50], OD-VVCA [15]) are typically discretization-based, leading inefficiency as the problem size (e.g., number of goods) increases, a phenomenon often termed the "curse of dimensionality" [3, 4].

Example 1.1 (Curse of Dimensionality)**.** *Consider that a seller wants to sell m goods to one buyer. The valuation of buyer is additive among goods, and her value for good i is* $v_i \overset{\text{i.d.}}{\sim} Bernoulli(p = 0.5)$[2]*. It's straight-forward that selling each good with price* $p_i \equiv 1, i \in [m]$ *can maximize the revenue of seller.*

Conventional approaches discretize the outcome space $([0,1]^m)$ *with K candidates* $\{x_1, ..., x_K\} \subseteq [0,1]^m$ *and prices to each candidates,* $\{p_1, ..., p_K\}$*. The j'th element in k'th candidate,* x_{kj}*, represents the probability to trade good j. There are* 2^m *deterministic outcomes* $(\{(a_1, ..., a_m) : a_i = 0 \text{ or } 1\})$ *in this problem, and all outcomes are indispensable to recover the optimal mechanism. Yet the optimal mechanism itself has a precise description, which can be* $\text{poly}(m)$*.*

Example 1.1 motivates our novel concept of *pricing rules*, as illustrated in Definition 1.2.

Definition 1.2 (Pricing Rule (Informal))**.** A pricing rule is a function $f : \mathcal{X} \to \mathbb{R}$ that assigns a price to each possible outcome. If the buyer selects the outcome $x \in \mathcal{X}$, then she shall pay $f(x)$.

Example 1.3. *Let* $X = [0,1]^m$ *be the outcome space in Example 1.1. Given the pricing rule f, the rational buyer with valuation* $v \in \mathbb{R}^m$

[1] Full version and codes can be found at Ma et al. [33].

[2] A Bernoulli random variable with $p = 0.5$ can be 0 or 1 with equal probability.

* Corresponding author.

Proc. of the 25th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2026), C. Amato, L. Dennis, V. Mascardi, J. Thangarajah (eds.), May 25 – 29, 2026, Paphos, Cyprus. © 2026 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). https://doi.org/10.65109/JTXL5273

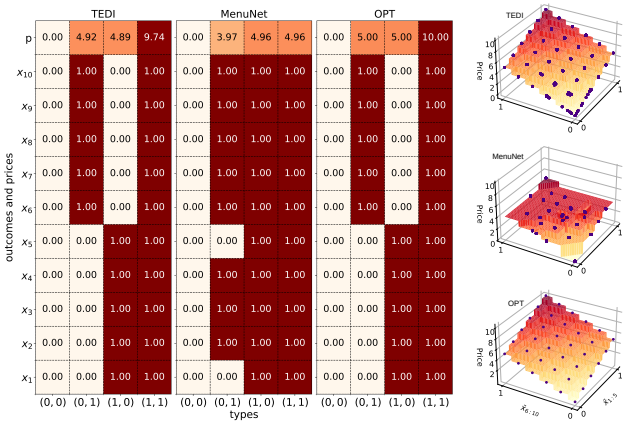

Figure 1: (a): A visualization of the optimal mechanism (OPT) and the mechanism learned by TEDI and MenuNet[44], in the $n = 1$ buyer, $m = 10$ goods setting with independent $Bernoulli(p = 0.5)$ valuation. The horizontal axis, (x, y), represents that the buyer's 10-dimensional valuation has the expression $(x, x, x, x, x, y, y, y, y, y) \in \{0, 1\}^{10}$. The $\{x_i\}_{1 \le i \le 10}$ (resp. p) in the vertical axis represents the probability of good i sold (resp. the price charged) to the buyer. (b): A visualization of the mechanisms given by TEDI, MenuNet, and OPT. Each point (x, y) represents one realized outcome $(a_1, ..., a_{10})$ with the mapping $x = \frac{1}{5}\sum_{1 \le i \le 5} a_i$ and $y = \frac{1}{5}\sum_{6 \le i \le 10} a_i$. The height value represents the price corresponding to each outcome. (Overall): TEDI exhibits $> 98\%$ optimal revenue, a contrast with 81.1% for MenuNet.

will choose the bundle $x \in X$ that maximizes her utility ($\langle v, x\rangle - f(x)$). In Example 1.1, optimal pricing rule has the expression $f(x) = \langle \mathbf{1}, x\rangle$, which naturally has an $O(m)$ representation.

1.1 Our Contributions

The main technical contribution of this paper is to establish an algorithm named TEDI (**T**ruthful, **E**xpressive, and **D**imension-**I**nsensitive approach, detailed in Section 3) with a suite of innovative techniques to learn (multi-player) optimal pricing rules. The core idea is to parameterize pricing rules using neural networks that possess universal approximation properties *w.r.t.* a "properly defined" function class. We also develop two novel techniques, namely *covariance trick* and *continuous sampling*, to efficiently sample an unbiased gradient estimator, which enables the learning process tractable by combining first-order algorithms (*e.g.*, Adam). TEDI exhibits several ideal properties established in Section 3.4: *Truthfulness*: Any mechanism learned by TEDI is truthful; *Full-Expressiveness*: TEDI can approximate the optimal mechanism with arbitrary precision; and *Dimension-Insensitivity*: TEDI maintains good performance as computational resources grow polynomially with the problem size. The *dimension-insensitivity* is further demonstrated through experiments in Section 4. Figure 1 illustrates a performance comparison among TEDI, optimal mechanisms, and existing approaches with setting described in Example 1.1, which showcases the superior performance of TEDI over discretization-based approaches.

2 PRELIMINARY

Consider a mechanism design model consisting of n participants (*e.g.*, bidders in an auction), one principal (*e.g.*, auctioneer or mechanism designer) and m kinds of goods which have potentially unlimited supply (*e.g.*, goods can be reproduced with certain costs). Each participant $i \in [n]$ owns a private type $t_i \in \mathcal{T}_i := [0, 1]^m$, where $\mathcal{T}_i$ is the type space of participant i and t_{ij} is the valuation of good j to participant i. Denote $\mathcal{T} = \times_{i \in [n]} \mathcal{T}_i$ as the space of type profile and $\boldsymbol{t} = (t_1, ..., t_n) \in \mathcal{T}$, $\boldsymbol{t}_{-i} = (t_1, ..., t_{i-1}, t_{i+1}, ..., t_n) \in \mathcal{T}_{-i}$ as the type profiles w/o participant i's type, respectively. Due to the private nature of individual types, the principal only knows the joint distribution over type profiles, denoted as $\mathcal{F} \in \Delta(\mathcal{T})$. The principal aims to design a direct mechanism that uses the participants' reported types $\boldsymbol{t}' = (t_i') \in \mathcal{T}$ as input and outputs an outcome $\boldsymbol{x} \in \mathcal{X} := [0, 1]^{n \times m}$ as well as a monetary transfer vector $\boldsymbol{p} = (p_1, \ldots, p_n) \in \mathbb{R}^n$. Here, $\mathcal{X}$ is the outcome space, x_{ij} is the probability of allocating one-unit good j to participant i, and p_i is the monetary transfer from participant i to the principal. Specially, each participant receives her local outcome $x_i \in \mathcal{X}_i := [0, 1]^m$, and $\boldsymbol{x} = (x_1, ..., x_n) \in \mathcal{X}$ is referred to as a global outcome. The utility of participant i, $u_i(x_i, p_i; t_i) : \mathcal{X}_i \times \mathbb{R} \times \mathcal{T}_i \to \mathbb{R}$, when her local outcome is x_i, the monetary transfer is p_i, and her type is t_i, is assumed to be quasi-linear, risk-neutral and additive among goods as considered in [9, 19, 50], and thus uniquely represented by the form: $u_i(x_i, p_i; t_i) = \langle x_i, t_i\rangle - p_i$. Additionally, we use $u_0(\boldsymbol{x}, \boldsymbol{p}; \boldsymbol{t}) : \mathcal{X} \times \mathbb{R}^n \times \mathcal{T} \to \mathbb{R}$ to denote the utility of the principal, relying on outcome $\boldsymbol{x}$, price vector $\boldsymbol{p}$ and type profile $\boldsymbol{t}$. We impose no special restrictions on $u_0(\boldsymbol{x}, \boldsymbol{p}; \boldsymbol{t})$ and leave the specific form to experiments. Meanwhile, we require that the principal is risk-neutral towards participants' type profile distribution $\mathcal{F}$ thus would like to maximize her expected utility, which is standard in micro-economics [35]. Given all above, a mechanism design problem $\mathcal{MD}$ is uniquely determined by a 4-tuple $\mathcal{MD} = (n, m, u_0, \mathcal{F})$.

Remark 2.1. *It is important to note that we do not impose the unit-supply constraint $\sum_{i \in [n]} x_{ij} \le 1, j \in [m]$ on outcome space $\mathcal{X}$ as in [9, 19, 50]. This is because in our model goods have potentially unlimited supply. Consequently, any solution of the form $(x_1, ..., x_n), x_i \in [0, 1]^m$ constitutes a feasible outcome.*

The main task of this work is to design an algorithm to output a truthful (IC or strategy-proof) and individual rational (IR) direct mechanism that maximizes the principal's expected utility given an instance $\mathcal{MD}$ as input.

Definition 2.2 (Direct Mechanisms)**.** A direct mechanism $M^d = (\boldsymbol{x}^d, \boldsymbol{p}^d)$[3] consists of an allocation rule $\boldsymbol{x}^d : \mathcal{T} \to \mathcal{X}$ and a payment rule $\boldsymbol{p}^d : \mathcal{T} \to \mathbb{R}^n$. Given a reported type profile $\boldsymbol{t}'$, a direct mechanism M^d implements the global outcome $\boldsymbol{x}^d(\boldsymbol{t}') \in \mathcal{X}$ and monetary transfer $\boldsymbol{p}^d(\boldsymbol{t}') \in \mathbb{R}^n$. For each participant i with true type t_i, she realizes the local outcome $x_i^d(\boldsymbol{t}')$ and the monetary transfer $p_i^d(\boldsymbol{t}')$, thus yielding utility $u_i(x_i^d(\boldsymbol{t}'), p_i^d(\boldsymbol{t}'); t_i)$.

Definition 2.3 (Truthful and IR Direct Mechanisms)**.** A direct mechanism $M^d = (\boldsymbol{x}^d, \boldsymbol{p}^d)$ is truthful, if reporting true type is a dominant strategy for each participant, regardless of other participants' reported types $\boldsymbol{t}'_{-i}$. Mechanism M^d is IR, if each participant

[3]The superscript d represents its role in "**d**irect mechanism".

is happier to participate in the mechanism instead of realizing the outside option ($x_i = \mathbf{0}, p_i = 0$, meaning receives nothing without payment) at any case. The properties of truthfulness and IR can be mathematically expressed by following two conditions for all $i \in [n]$, participant i's type $t_i \in \mathcal{T}_i$ and participant $-i$'s reported types $\boldsymbol{t}'_{-i} \in \mathcal{T}_{-i}$:

$$u_i(x_i^d(t_i, \boldsymbol{t}'_{-i}), p_i^d(t_i, \boldsymbol{t}'_{-i}); t_i) \geq 0, \quad \text{(IR)}$$

$$u_i(x_i^d(t_i, \boldsymbol{t}'_{-i}), p_i^d(t_i, \boldsymbol{t}'_{-i}); t_i) \geq u_i(x_i^d(t'_i, \boldsymbol{t}'_{-i}), p_i^d(t'_i, \boldsymbol{t}'_{-i}); t_i). \quad \forall t'_i \in \mathcal{T}_i \quad \text{(IC)}$$

From now on, we refer truthful and IR direct mechanisms as truthful mechanisms for simplicity when the context allows. We denote $\mathcal{M}^d$ as the set of all direct mechanisms. Formally, the goal *w.r.t.* a mechanism design problem $\mathcal{MD}$ can be stated as solving the Program (1).

$$\max_{M^d \in \mathcal{M}^d} \ \mathrm{EU}(M^d) \coloneqq \mathbb{E}_{\boldsymbol{t} \sim \mathcal{F}} \left[u_0(\boldsymbol{x}^d(\boldsymbol{t}), \boldsymbol{p}^d(\boldsymbol{t}); \boldsymbol{t}) \right]. \ \text{s.t.}(IC), (IR) \quad (1)$$

In Section 3, we will introduce the algorithm, referred to as TEDI, that solves Program (1).

2.1 Characterization Results

Prior to elaborating on TEDI, we begin with introducing fundamental concepts related to pricing rules and a pivotal characterization result, both of which are essential for understanding TEDI.

Definition 2.4 (Pricing Rule)**.** Fix a problem instance $\mathcal{MD}$, a pricing rule *w.r.t.* participant i is a function $p_i^m : \mathcal{X}_i \times \mathcal{T}_{-i} \to \mathbb{R}$. [4]

Pricing rules admit following economic interpretation: Fix the type profile of the other participants, $\boldsymbol{t}_{-i}$. If participant i chooses a local outcome x_i, she shall pay $p_i^m(x_i; \boldsymbol{t}_{-i})$ to the principal. Note that pricing rules differ from payment rules in direct mechanisms on their respective domain ($\mathcal{X}_i$ v.s. $\mathcal{T}_i$).

Definition 2.5 (Menu Mechanisms)**.** A menu mechanism M^m consists of n pricing rules, *i.e.*, $M^m = \{p_i^m\}_{i \in [n]}$, where $p_i^m : \mathcal{X}_i \times \mathcal{T}_{-i} \to \mathbb{R}$ is the pricing rule *w.r.t.* participant i.

Given a type profile $\boldsymbol{t}$, the menu mechanism M^m determines the optimal local outcome $x_i^* \in \mathcal{X}_i$ for each participant i that maximizes her utility, i.e., $x_i^* \in \mathrm{argmax}_{x_i \in \mathcal{X}_i} u_i(x_i, p_i^m(x_i, \boldsymbol{t}_{-i}); t_i)$[5]. The corresponding monetary transfer $p_i^* \in \mathbb{R}$ for participant i is then defined as the price associated with x_i^*: $p_i^* = p_i^m(x_i^*; \boldsymbol{t}_{-i})$. The resulting global outcome and monetary transfers are respectively given by, $\boldsymbol{x}^* = (x_1^*, ..., x_n^*)$ and $\boldsymbol{p}^* = (p_1^*, ..., p_n^*)$.

Unlike conventional menu mechanisms (as in Example 1.1) that enumerate a set of candidates to represent the outcome space, the menu mechanisms here are represented by pricing rules, making them discretization-free. This approach allows us to express menu mechanisms more efficiently and to analyze them from a functional perspective, as shown in Theorem 2.9. Before that, we introduce some definitions first.

Definition 2.6 (Equivalence Between Mechanisms)**.** Let $M^m = \{p_i^m\}_{i \in [n]}$ and $M^d = (\boldsymbol{x}^d, \boldsymbol{p}^d)$ be a menu mechanism and a direct mechanism, respectively. We say M^m and M^d are equivalent if following holds for all type profiles $\boldsymbol{t} \in \mathcal{T}$,

$$x_i^d(\boldsymbol{t}) \in \mathrm{argmax}_{x_i \in \mathcal{X}_i} u_i(x_i, p_i^m(x_i; \boldsymbol{t}_{-i}); t_i), \ \ p_i^d(\boldsymbol{t}) = p_i^m(x_i^d(\boldsymbol{t}); \boldsymbol{t}_{-i}).$$

Note that in Definition 2.6, if M^d is equivalent with M^m, then given the same type profile $\boldsymbol{t}$, the outcome as well as the monetary transfer of M^d are exactly those of menu mechanism M^m. We next define two conditions for constructing the equivalence between menu mechanisms and truthful direct mechanisms.

Definition 2.7 (No-buy-no-pay)**.** $M^m = \{p_i^m\}_{i \in [n]}$ satisfies no-buy-no-pay, if $\forall i \in [n], \boldsymbol{t}_{-i} \in \mathcal{T}_{-i}$, we have $p_i^m(\mathbf{0}, \boldsymbol{t}_{-i}) \leq 0$.

Definition 2.8 (Partial-convexity)**.** $M^m = \{p_i^m\}_{i \in [n]}$ satisfies partial convexity, if $\forall i \in [n], \boldsymbol{t}_{-i} \in \mathcal{T}_{-i}$, $p_i^m(x_i; \boldsymbol{t}_{-i})$ is convex on x_i.

The definition *partial convexity* comes from that the *convexity* of $p_i^m(x_i, \boldsymbol{t}_{-i})$ is required on *partial* inputs (x_i). Partial convexity has multiple consequences: one is establishing the expressiveness power of pricing rules (Theorem 2.9), the other is guaranteeing the concavity of participants' utilities, consequently making inference of menu mechanism tractable through convex program (Equation (4)). Following theorem characterizes the equivalence between menu mechanisms and truthful direct mechanisms. The proof of Theorem 2.9 (as well as other theorems) can be found in full version.

Theorem 2.9. (Equivalence Between Mechanism Classes)
*Let $\mathcal{M}^{d,t}$ be the class of all **truthful** direct mechanisms and $\mathcal{M}^{m,pn}$ be the class of all menu mechanisms satisfying **partial convexity** and **no-buy-no-pay**. Then, $\mathcal{M}^{d,t}$ and $\mathcal{M}^{m,pn}$ are equivalent in the following sense:*
- *For any truthful direct mechanism $M^d \in \mathcal{M}^{d,t}$, there is a menu mechanism $M^m \in \mathcal{M}^{m,pn}$ such that M^m is equivalent with M^d.*
- *For any menu mechanism $M^m \in \mathcal{M}^{m,pn}$, and for any direct mechanism $M^d \in \mathcal{M}^d$ that is equivalent with M^m, it holds that $M^d \in \mathcal{M}^{d,t}$ (*i.e.*, M^d is truthful).*

3 DESIGN OF TEDI

This section introduces the design of TEDI, organized around three components—parameterization, inference, and training (see Section 3.1 to 3.3). The whole procedure is illustrated in Figure 2. Pseudocodes for the training procedure are provided and explained at the end of Section 3.

Prior to delving into the technical details of TEDI, we first outline the key challenges in its designing and our high-level ideas to address them.

Challenge 1: Parameterizing Pricing Rules. Theoretical equivalence (Theorem 2.9) motivates us to parameterize pricing rules using neural networks, explicitly encoding partial convexity and no-buy-no-pay constraints through architectural design. A critical challenge is that the network should be a universal approximator of partial convex functions in order to capture the optimal mechanism. However, such a network-based universal approximator of partial convex functions is unknown to our knowledge.

Challenge 2: Optimization on Pricing Rules. Unlike previous works, the inference of menu mechanism here necessitates computing argmax points in continuous space ($\mathcal{X}_i$), rather than discrete

[4] The superscript m represents its role in "**m**enu mechanism".
[5] When there might be multiple maximum points, any tie-breaking rule is feasible.

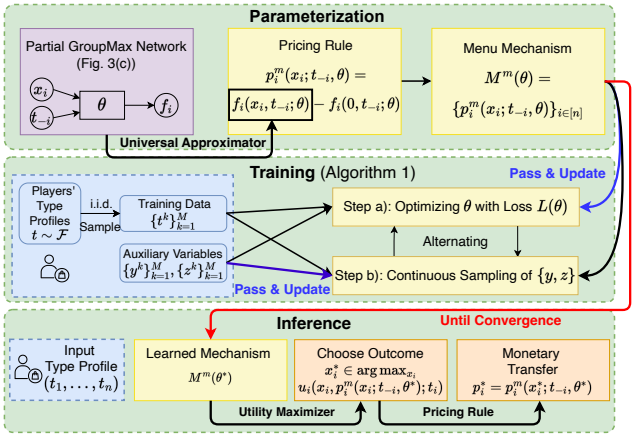

Figure 2: An illustration of procedure of TEDI. Best viewed in color (same for the remaining figures).

space. This shift leads to conventional "softmax" relaxation methods [9, 44, 50] intractable in optimization of network parameters.

Our solution leverages the insight that parameter updates do not necessitate direct computation of the objective function. Instead, we only require an unbiased gradient estimator and a stream of samples from the underlying distribution. To achieve this purpose, we develop two innovative techniques—*covariance trick* and *continuous sampling*—which together effectively address this challenge.

3.1 Parameterization

Let $f_i(x_i, \boldsymbol{t}_{-i}; \theta)$ denote a neural network (specifically, Partial GroupMax Network) represented by parameters θ, taking $(x_i, \boldsymbol{t}_{-i})$ as input and output $f_i(x_i, \boldsymbol{t}_{-i}; \theta)$. The network parameter is shared among all participants $i \in [n]$. The pricing rule for participant i, p_i^m, is represented in the following expression,

$$p_i^m(x_i; \boldsymbol{t}_{-i}, \theta) = f_i(x_i, \boldsymbol{t}_{-i}; \theta) - f_i(\mathbf{0}, \boldsymbol{t}_{-i}; \theta). \tag{2}$$

Note that no-buy-no-pay is satisfied from Equation (2) automatically. To ensure the partial convexity of pricing rules, it suffices to enforce partial convexity of $f_i(x_i; \boldsymbol{t}_{-i}, \theta)$ solely on x_i. We subsequently design *Partial GroupMax Network*, a universal approximator of partial convex functions, to parameterize $f_i(x_i, \boldsymbol{t}_{-i}; \theta)$. The design of Partial GroupMax Network is inspired by GroupMax Network [51], PMA [29] and PICNN [2]. Before formalizing Partial GroupMax Network, we introduce two foundational sub-components: GroupMax activation and Parameterized Affine Network (PAN).

Definition 3.1 (GroupMax Activation)**.** The GroupMax activation function has the expression:

$$\text{GroupMax} : \mathbb{R}^{G \times E} \to \mathbb{R}^G \qquad \text{GroupMax}(x)_i = \max_{1 \le j \le E} x_{i,j}. \tag{3}$$

The intuition behind Equation (3) is below: The GroupMax activation function equally distributes the $G \cdot E$ elements into G groups, with each group containing E elements. It then outputs a vector that collects the largest element from each group. Figure 3-(a) illustrates the GroupMax activation function.

Definition 3.2 (Parameterized Affine Network)**.** A Parameterized Affine Network is an MLP (Multi-Layer Perceptron) that parameterizes functions $W : \mathbb{R}^{d_y} \to \mathbb{R}^{d^{out} \times d^{in}}$ and $b : \mathbb{R}^{d_y} \to \mathbb{R}^{d^{out}}$, where

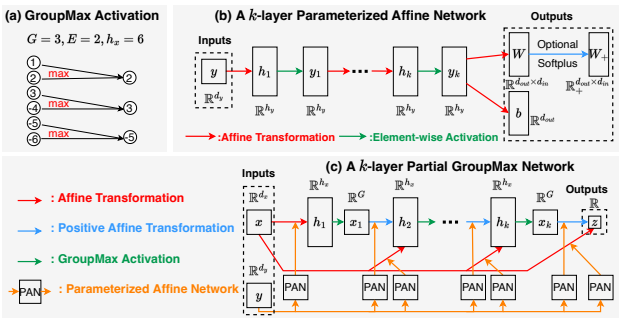

Figure 3: The architecture of a k-layer Partial GroupMax Network and its sub-components.

d_y is the input dimension of Parameterized Affine Network, d^{in} and d^{out} are the dimensions of input vectors and output vectors of an affine transformation.

Given an input y, a Parameterized Affine Network produces parameters $W(y) \in \mathbb{R}^{d^{out} \times d^{in}}$ and $b(y) \in \mathbb{R}^{d^{out}}$, which can perform an affine transformation on a vector $z \in \mathbb{R}^{d^{in}}$ as: $W(y) \cdot z + b(y)$. Sometimes $W(y)$ is required to be positive, then we implement an element-wise softplus function[6] to ensure the positivity. The architecture of a k-layer Parameterized Affine Network is illustrated in Figure 3-(b). Next, let us define Partial GroupMax Network based on the above components.

Definition 3.3 (Partial GroupMax Network)**.** A Partial GroupMax Network parameterizes a function $f : \mathbb{R}^{d_x} \times \mathbb{R}^{d_y} \to \mathbb{R}$, such that $f(x, y)$ is convex on x and continuous on (x, y) given inputs $x \in \mathbb{R}^{d_x}, y \in \mathbb{R}^{d_y}$.

Compared with an MLP, Partial GroupMax Network has two key differences: (1) all activation functions are replaced by GroupMax activation, which is no longer element-wise; and (2) all affine transformations in Partial GroupMax Network are generated by a series of Parameterized Affine Networks that take y as input.

Specifically, a K-layer Partial GroupMax Network first generates all affine transformations $\{W_k(y), b_k(y)\}_{k=0}^K$ with hardcoded constraints $W_k(y) \ge 0, \forall k \ge 1$, which are output by Parameterized Affine Network on input y. Given these affine transformations, the remaining computation process on x is following: $h_k(x, y) = W_{k-1}(y)x_{k-1}(x, y) + b_{k-1}(y)$ and $x_k(x, y) = \text{GroupMax}(h_k(x, y))$ for $k = 1, 2, \ldots, K$, where $x_0(x, y) = x$. Partial GroupMax Network finally outputs $z(x, y) = W_K(y)x_K(x, y) + b_K(y)$.[7] An illustration of a k-layer Partial GroupMax Network is provided in Figure 3-(c).

We highlight the following key property of Partial GroupMax Network, which plays a crucial role in TEDI.

Proposition 3.4. *Partial GroupMax Network is a universal approximator of all functions $f(x, y)$, which are convex on x and continuous on (x, y).*

[6] A softplus function has the expression: $\text{softplus}(x) = \log(1 + \exp(x))$. It maps a real number to a positive number.

[7] We allow residual connections in Partial GroupMax Network but omit here for simplicity.

Proposition 3.4 has a twofold meaning: On one hand, any instance of Partial GroupMax Network, denoted as $f(x, y)$, is guaranteed to be convex on x and continuous on (x, y). On the other hand, any function that is convex on x and continuous on (x, y) can be arbitrarily approximated by Partial GroupMax Network.

3.2 Inference

The task of *inference* component of TEDI, is to compute the outcome $\boldsymbol{x}^* = (x_1^*, \cdots, x_n^*)$ and monetary transfer $\boldsymbol{p}^*$ given type profile $\boldsymbol{t}$ and parameter θ. Specifically, each local outcome x_i^* is derived by maximizing participant i's utility *w.r.t.* pricing rule $p_i^m(x_i; \boldsymbol{t}_{-i}, \theta)$, and p_i^* is determined by $p_i^* = p_i^m(x_i^*; \boldsymbol{t}_{-i}, \theta)$. The utility-maximization problem for participant i is formalized as follows:

$$\begin{aligned} \max_{x_i \in \mathcal{X}_i} \quad & u_i(x_i, p_i^m(x_i; \boldsymbol{t}_{-i}, \theta); t_i) = \langle t_i, x_i \rangle - p_i^m(x_i; \boldsymbol{t}_{-i}, \theta) \\ & = \langle t_i, x_i \rangle - f_i(x_i; \boldsymbol{t}_{-i}, \theta) + f_i(\boldsymbol{0}; \boldsymbol{t}_{-i}, \theta). \end{aligned} \tag{4}$$

Note that the first term, $\langle t_i, x_i \rangle$, is linear on x_i; the second term, $-f_i(x_i; \boldsymbol{t}_{-i}, \theta)$, is concave on x_i; and the third term, $f_i(\boldsymbol{0}; \boldsymbol{t}_{-i}, \theta)$, is constant *w.r.t.* x_i. Overall, the objective function in (4) is concave on x_i. Finding the maximum point x_i^* of a concave function with convex constraints $x_i \in \mathcal{X}_i$ is tractable through convex optimization [6], and the corresponding p_i^* can be derived naturally.

3.3 Training

Training is the key component of TEDI, and this subsection provides a detailed introduction to it at length. For simplicity, we use $u_0(\boldsymbol{x}, \theta; \boldsymbol{t})$ and $u_i(x_i; \boldsymbol{t}, \theta)$ to represent $u_0(\boldsymbol{x}, \{p_i(x_i; \boldsymbol{t}_{-i}, \theta)\}_{i\in[n]}; \boldsymbol{t})$ and $u_i(x_i, p_i(x_i; \boldsymbol{t}_{-i}, \theta); t_i)$, as the utilities of principal and participants depend on outcome $\boldsymbol{x}$, type profile $\boldsymbol{t}$ and menu mechanism parameterized by θ. (Note that the prices $\{p_i(x_i; \boldsymbol{t}_{-i}, \theta)\}_{i\in[n]}$ are then uniquely determined by θ, $\boldsymbol{x}$, and $\boldsymbol{t}$.)

Denote $\boldsymbol{x}(\boldsymbol{t})$ as the outcome induced by type profile $\boldsymbol{t}$. Fix θ, the corresponding local outcome $x_i(\boldsymbol{t})$ must be subject to the participants' utility maximization constraints, *i.e.*, $\forall i \in [n]$, $x_i(\boldsymbol{t}) \in \operatorname{argmax}_{a_i \in \mathcal{X}_i} u_i(a_i; \boldsymbol{t}, \theta)$. The principal's utility is then determined by $u_0(\boldsymbol{x}(\boldsymbol{t}), \theta; \boldsymbol{t})$. Because the type profiles are drawn from $\mathcal{F}$, the principal would like to maximize the expected utility, shown as (5):

$$\begin{aligned} \max_{\theta, \boldsymbol{x}(\boldsymbol{t})} \quad & \mathbb{E}_{\boldsymbol{t}\sim\mathcal{F}}[u_0(\boldsymbol{x}(\boldsymbol{t}), \theta; \boldsymbol{t})] \\ \text{s.t.} \quad & x_i(\boldsymbol{t}) \in \operatorname{argmax}_{a_i \in \mathcal{X}_i} u_i(a_i; \boldsymbol{t}, \theta) \quad \forall i \in [n], \forall \boldsymbol{t} \in \mathcal{T} \end{aligned} \tag{5}$$

In the remainder of Section 3.3, we will explain how TEDI solves Program (5) step-by-step.

3.3.1 Simplifying the Program. The first difficulty is that Program (5) consists of infinite constraints and the objective function has an expectation form. To address this issue, we begin by sampling M i.i.d. type profiles from the distribution $\mathcal{F}$, denoted as $\mathcal{T}^M = \{\boldsymbol{t}^1, ..., \boldsymbol{t}^M\}$. Thus, we can simplify the objective and constraints by using a sample-average approximation. Here, Program (5) becomes

$$\begin{aligned} \max_{\theta, \{\boldsymbol{x}^k\}_{k\in[M]}} \quad & \frac{1}{M}\sum_{k=1}^{M}\left[u_0(\boldsymbol{x}^k, \theta; \boldsymbol{t}^k)\right] \\ \text{s.t.} \quad & x_i^k \in \operatorname{argmax}_{a_i \in \mathcal{X}_i} u_i(a_i; \boldsymbol{t}^k, \theta), \quad \forall i \in [n], \forall k \in [M] \end{aligned} \tag{6}$$

As $M \to \infty$, Program (6) approaches the original program (5).

3.3.2 Softening the Constraints. The second difficulty is that the computation process of the argmax operator in Program (6) is non-differentiable. To resolve this difficulty, we relax the argmax operator as a "softmax-like" distribution. Specifically, we construct the (energy-based) distribution $x_i \sim q_i^{\beta_i}(\cdot; \boldsymbol{t}, \theta)$, where

$$q_i^{\beta}(a_i; \boldsymbol{t}, \theta) = \frac{\exp\left(\beta \cdot u_i(a_i; \boldsymbol{t}, \theta)\right)}{Z_i^{\beta}(\boldsymbol{t}, \theta)}, \tag{7}$$

$$Z_i^{\beta}(\boldsymbol{t}, \theta) = \int_{a_i \in \mathcal{X}_i} \exp\left(\beta \cdot u_i(a_i; \boldsymbol{t}, \theta)\right) \mathrm{d}a_i, \tag{8}$$

$\beta > 0$ is the temperature and $Z_i^{\beta}(\boldsymbol{t}, \theta)$ is the normalizing constant. It's clear to see that $q_i^{\beta}(\cdot; \boldsymbol{t}, \theta)$ is indeed a distribution since $\int_{\mathcal{X}_i} q_i^{\beta}(a_i; \boldsymbol{t}, \theta)\mathrm{d}a_i \equiv 1$. Program (6) then becomes

$$\max_{\theta} \quad \mathrm{OBJ}^{\boldsymbol{\beta}}(\theta) = \frac{1}{M}\sum_{k=1}^{M} \mathbb{E}_{x_i^k \overset{\text{i.d.}}{\sim} q_i^{\beta_i}(\cdot; \boldsymbol{t}^k, \theta)}\left[u_0(\boldsymbol{x}^k, \theta; \boldsymbol{t}^k)\right]. \tag{9}$$

By $\overset{\text{i.d.}}{\sim}$ we mean the variables are independently distributed. Note that when $\beta_i \to \infty$, the distribution $q_i^{\beta_i}(\cdot; \boldsymbol{t}, \theta)$ approximates the point mass at $\{x_i : x_i \in \operatorname{argmax}_{a_i \in \mathcal{X}_i} u_i(a_i; \boldsymbol{t}, \theta)\}$. Therefore, Program (9) approaches Program (6) as all $\beta_i \to \infty$.

3.3.3 Differentiating over θ with Covariance Trick. From now on, we focus on the latest Program (9), which takes the form of an unconstrained optimization program on θ with objective function $\mathrm{OBJ}^{\boldsymbol{\beta}}(\theta)$. Following conventions in optimization and machine learning [5], our next step is to find an unbiased estimator of $\nabla_\theta \mathrm{OBJ}^{\boldsymbol{\beta}}(\theta)$, namely the first-order gradient of the objective. Given this estimator, θ can be optimized by the downstream first-order algorithm.

One might think $\nabla_\theta \mathrm{OBJ}^{\boldsymbol{\beta}}(\theta)$ is intractable, since θ appears not only in the utility function u_0 but also under the expectation operator ($\mathbb{E}_{x_i^k \overset{\text{i.d.}}{\sim} q_i^{\beta_i}(\cdot; \boldsymbol{t}^k, \theta)}[\cdot]$), making ∇_θ and $\mathbb{E}[\cdot]$ non-interchangeable. Surprisingly, following identity in Proposition 3.5 holds universally, enabling the interchange of ∇_θ with the expectation operator.

Proposition 3.5. (Covariance Trick) *Following identity holds universally.*

$$\begin{aligned} \nabla_\theta \mathbb{E}_{x_i}\left[u_0(\boldsymbol{x}, \theta; \boldsymbol{t})\right] &= \mathbb{E}_{x_i}\left[\nabla_\theta u_0(\boldsymbol{x}, \theta; \boldsymbol{t})\right] \\ &+ \mathrm{Cov}_{x_i}\left[u_0(\boldsymbol{x}, \theta; \boldsymbol{t}), \nabla_\theta \mathrm{ASW}^{\boldsymbol{\beta}}(\boldsymbol{x}, \boldsymbol{t}; \theta)\right], \end{aligned} \tag{10}$$

where the expectation is taken on $x_i \overset{\text{i.d.}}{\sim} q_i^{\beta_i}(\cdot; \boldsymbol{t}, \theta)$, $\mathrm{ASW}^{\boldsymbol{\beta}}(\boldsymbol{a}, \boldsymbol{t}; \theta) = \sum_{i\in[n]} \beta_i u_i(a_i, \boldsymbol{t}; \theta)$ is the affine social welfare of participants under outcome $\boldsymbol{a} \in \mathcal{X}$, weighted by $\boldsymbol{\beta}$. $\mathrm{Cov}[X, Y] = \mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$ is the covariance of random variables (X, Y).

We then replace covariance $\mathrm{Cov}[\cdot]$ with expectation $\mathbb{E}[\cdot]$ and consequently transform the RHS of Equation (10) to following form:

$$\begin{aligned} \nabla_\theta \mathbb{E}_{x_i}\left[u_0(\boldsymbol{x}, \theta; \boldsymbol{t})\right] = \frac{1}{2}\mathbb{E}_{\{y_i, z_i\}_{i\in[n]}}\Big[&\nabla_\theta\left(u_0(\boldsymbol{y}, \theta; \boldsymbol{t}) + u_0(\boldsymbol{z}, \theta; \boldsymbol{t})\right) \\ &+ \left(u_0(\boldsymbol{y}, \theta; \boldsymbol{t}) - u_0(\boldsymbol{z}, \theta; \boldsymbol{t})\right) \cdot \nabla_\theta\left(\mathrm{ASW}^{\boldsymbol{\beta}}(\boldsymbol{y}; \boldsymbol{t}, \theta) - \mathrm{ASW}^{\boldsymbol{\beta}}(\boldsymbol{z}; \boldsymbol{t}, \theta)\right)\Big], \end{aligned} \tag{11}$$

where the expectation is taken on $\{y_i, z_i\} \overset{\text{i.d.}}{\sim} q_i^{\beta_i}(\cdot; \boldsymbol{t}, \theta)$.

3.3.4 Deriving the Loss Function. To obtain the gradient in Equation (11), we construct following real-valued loss function:

$$\begin{aligned} L^{\boldsymbol{\beta}}(\theta; \boldsymbol{t}) = \frac{1}{2}\mathbb{E}\Big[&\left(u_0(\boldsymbol{y}, \theta; \boldsymbol{t}) + u_0(\boldsymbol{z}, \theta; \boldsymbol{t})\right) \\ &+ \left(\tilde{u}_0(\boldsymbol{y}, \theta; \boldsymbol{t}) - \tilde{u}_0(\boldsymbol{z}, \theta; \boldsymbol{t})\right)\left(\mathrm{ASW}^{\boldsymbol{\beta}}(\boldsymbol{y}; \boldsymbol{t}, \theta) - \mathrm{ASW}^{\boldsymbol{\beta}}(\boldsymbol{z}; \boldsymbol{t}, \theta)\right)\Big], \end{aligned} \tag{12}$$

where $\tilde{u}_0$ indicates that this term does not participate in gradient computation. It can be implemented by '.detach()' module in PyTorch on the term $u_0(\boldsymbol{y}, \theta; \boldsymbol{t}) - u_0(\boldsymbol{z}, \theta; \boldsymbol{t})$.

Until now, the only task left for obtaining an unbiased estimator of $\nabla_\theta \mathrm{OBJ}^{\boldsymbol{\beta}}(\theta)$ is to draw $y_i^k, z_i^k \sim q_i^{\beta_i}(\cdot; \boldsymbol{t}^k, \theta)$ for each sample index $k \in [M]$, participant index $i \in [n]$ and time-contingent parameter θ. One would imagine drawing samples after each network iteration using existing techniques [20, 28, 30, 36]. However, re-drawing samples from scratch for every iteration is costly. The next paragraph introduces how we utilize the "continuous sampling" technique to draw these samples efficiently during full training period.

3.3.5 Continuous Sampling *of $y_i \sim q_i^{\beta_i}(\cdot; \boldsymbol{t}, \theta)$ for time-contingent θ.* Without loss of generality, we focus on continuous sampling of $y_i \sim q_i^{\beta_i}(\cdot; \boldsymbol{t}, \theta)$ for single participant i and one type profile $\boldsymbol{t}$. Denote θ^{last} and θ^{new} as the last-iteration and current network parameters, respectively. Assume that we already have an approximate sample y_i^{last} from $q_i^{\beta_i}(\cdot; \boldsymbol{t}, \theta^{last})$. We construct a Euler–Maruyama discretization of Langevin dynamic [42] with initial point $y_i^0 = y_i^{last}$ and iteration time T to output y_i^T as follows:

$$y_i^t = y_i^{t-1} + \eta_i \nabla_{x_i} u_i(y_i^{t-1}; \boldsymbol{t}, \theta^{new}) + \sqrt{2\eta_i \beta_i^{-1}}\epsilon_i^t,\ \epsilon_i^t \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1), \quad (13)$$

where $\eta_i > 0$ is the step size or learning rate. Notice that the distribution of y_i^{last} is close to $q_i^{\beta_i}(\cdot; \boldsymbol{t}, \theta^{new})$ (since θ^{new} is close to θ^{last}), and the distribution $q_i^{\beta_i}(\cdot; \boldsymbol{t}, \theta^{new})$ is log-concave. Thus, by nature of Langevin dynamics, the distribution of $\{y_i^t\}$ converges quickly to $q_i^{\beta_i}(\cdot; \boldsymbol{t}, \theta^{new})$ [42] and it's natural to take y_i^T as an approximate sample from $q_i^{\beta_i}(\cdot; \boldsymbol{t}, \theta^{new})$.

3.3.6 Training Procedure. Algorithm 1 presents the pseudo-codes of our training procedure. The training procedure begins with sampling M i.i.d. type profiles ($\{\boldsymbol{t}^k\}_{k\in[M]}$) as well as initializing auxiliary variables ($\{\boldsymbol{y}^k, \boldsymbol{z}^k\}_{k\in[M]}$) *w.r.t.* type profiles. In the training process, the algorithm alternates between (a) optimizing mechanism parameter θ with samples $\{\boldsymbol{y}^k, \boldsymbol{z}^k\}_{k\in[M]}$ (from line 5-7) and (b) continuous sampling of $\{\boldsymbol{y}^k, \boldsymbol{z}^k\}_{k\in[M]}$ as θ varies (from line 8-14).

To be specific, (a) In line 5-7, we regard $\boldsymbol{y}_i^k$ and $\boldsymbol{z}_i^k$ as approximate samples from $q_i^{\beta_i}(\cdot; \boldsymbol{t}^k, \theta)$. If these samples are exactly drawn from $q_i^{\beta_i}(\cdot; \boldsymbol{t}^k, \theta)$, then the expectation of $\nabla_\theta L(\theta; \boldsymbol{t}^k)$ in line 6 is exactly the true gradient. Therefore, we can regard $\nabla_\theta L(\theta; \boldsymbol{t}^k)$ taken on samples $\{\boldsymbol{y}^k, \boldsymbol{z}^k\}_{k\in[M]}$ as an approximate unbiased-gradient of the objective function in Equation (9). (b) In line 8-14, since we require the distribution of $\boldsymbol{y}_i^k$ and $\boldsymbol{z}_i^k$ close to $q_i^{\beta_i}(\cdot; \boldsymbol{t}^k, \theta)$, yet θ is continuously updated in step (a). Therefore, we also need to update $\{\boldsymbol{y}^k, \boldsymbol{z}^k\}_{k\in[M]}$ after each iteration of θ, to guarantee our requirements throughout the full training process.

3.4 Theoretical Justifications of TEDI

TEDI preserves several ideal properties, respectively: truthfulness, full expressiveness, and dimension-insensitivity. Specially, truthfulness of TEDI is a direct corollary of Theorem 2.9, as stated in Corollary 3.6.

Corollary 3.6. (TEDI is Truthful) *For any menu mechanism M^m that can be expressed by TEDI, and any direct mechanism M^d that is equivalent with M^m, it holds that M^d is truthful.*

Algorithm 1: Training of TEDI

Input: $\mathcal{MD} = (n, m, u_0, \mathcal{F})$: a mechanism design problem
Output: θ: parameter of menu mechanisms

1 **parameter**: M: number of samples, S: iteration in Langevin dynamics, T: total training iteration, β_0: initial temperature, η_0: initial step size in Langevin dynamics, η_θ: initial learning rate for θ
2 Sample M type profiles $\{\boldsymbol{t}^k\}_{k\in[M]} \overset{\text{i.d.}}{\sim} \mathcal{F}(\mathcal{T})$, initialize $\{\boldsymbol{y}^k, \boldsymbol{z}^k\}_{k\in[M]} \overset{\text{i.d.}}{\sim} \mathcal{U}(\mathcal{X})$;
3 Initialize $\beta_i \leftarrow \beta_0, \eta_i \leftarrow \eta_0, \forall i$, and learning rate η_θ;
/* Training Loop */
4 **for** $t = 1, 2, ..., T$ **do**
/* a) Updating mechanism θ by an approximate unbiased-estimator of true gradient */
5 Compute $L^{\boldsymbol{\beta}}(\theta; \boldsymbol{t}^k)$ in Equation (12) for all $k \in [M]$, with $\boldsymbol{y}, \boldsymbol{z}$ in (12) replaced by $\boldsymbol{y}^k, \boldsymbol{z}^k$;
6 $L(\theta) \leftarrow -\frac{1}{M}\sum_{k\in[M]} L^{\boldsymbol{\beta}}(\theta; \boldsymbol{t}^k)$;
7 Compute $\nabla_\theta L(\theta)$ by '.backward()' and optimize θ
/* b) Resampling $\{\boldsymbol{y}^k, \boldsymbol{z}^k\}_{k\in[M]}$ by Langevin Dynamic */
8 **for** $k = 1, 2, ..., M; i = 1, 2, ..., n$ **do**
9 $y_i^{k,0} \leftarrow y_i^k, z_i^{k,0} \leftarrow z_i^k$;
10 **for** $s = 1, 2, ..., S$ **do**
11 $y_i^{k,s} = y_i^{k,s-1} + \eta_i \cdot \nabla_{x_i} u_i(y_i^{k,s-1}; \boldsymbol{t}, \theta) + \sqrt{2\eta_i \beta_i^{-1}} \cdot \epsilon_i^{k,s}$ where $\epsilon_i^{k,s} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$, do the same on $z_i^{k,s}$.
12 **end**
13 $y_i^k \leftarrow y_i^{k,S}, z_i^k \leftarrow z_i^{k,S}$;
14 **end**
15 Increase $\{\beta_i\}_{i\in[n]}$ as well as decrease $\{\eta_i\}_{i\in[n]}$ and η_θ when appropriate;
16 **end**
17 **return**: Learned menu mechanism θ;

Given the results established in Theorem 2.9 and Proposition 3.4, the insight of full-expressiveness then becomes clear: On one hand, truthful mechanisms are equivalent to menu mechanisms with partial convex pricing rules. Subsequently, partial convex pricing rules can be arbitrarily approximated by Partial GroupMax Network. Following above insight, we (informally) state the *full-expressiveness* in Proposition 3.7. Readers can find a formal statement in full version.

Proposition 3.7 (TEDI is Full-Expressive (Informal))**.** *Fix a problem instance $\mathcal{MD}$. Denote* SEU^* (SEU^{TEDI}) *as the principal's supreme expected utility over all truthful mechanisms (mechanisms expressed by TEDI)* w.r.t. $\mathcal{MD}$*, then,* $\mathrm{SEU}^* = \mathrm{SEU}^{TEDI}$.

Lastly, we provide a *dimension-insensitivity* justification. Following the machine learning convention, it's widely believed that neural networks are ideal approaches to overcome the "curse of dimensionality", because network architecture has the potential to discover simple structures [25, §6.4]. As TEDI utilizes network structure to parameterize pricing rules, we believe that TEDI is

more dimension-insensitive than discretization-based approach to learn optimal pricing rules.

4 EXPERIMENTS

In this section, we conduct thorough experiments to show the competitive performance of TEDI .

4.1 Experimental Settings

In our experiments, the problem $\mathcal{MD}$ is economically interpreted as reproducible goods auctions [1, 8, 27], where principal commits to produce goods with one-time production cost $c^p \in \mathbb{R}_+$ and duplication cost per-unit $c^d \in \mathbb{R}_+$ to satisfy participants' demand. Principal's utility is $u_0(\boldsymbol{x}, \boldsymbol{p}, \boldsymbol{t}) = \sum_{i\in[n]} p_i - c_0(\boldsymbol{x})$, where $c_0(\boldsymbol{x}) = \sum_{j\in[m]} \left(c^p \cdot \max_{i\in[n]} x_{ij} + c^d \cdot \sum_{i\in[n]} x_{ij}\right)$ is the cost of implementing the outcome $\boldsymbol{x}$ by production.

$c_0(\boldsymbol{x})$ has following explanations: For each good j, since participant i buys the good j with probability x_{ij}, good j should be produced with probability at least x_{ij}. This argument is universal for i, thus good j should be produced with probability at least $\max_{i\in[n]} x_{ij}$. Also note that producing good j with probability $\max_{i\in[n]} x_{ij}$ is feasible to implement global outcome $\boldsymbol{x}$. Therefore, the principal would like to produce good j with probability $\max_{i\in[n]} x_{ij}$, with expected one-time production cost $c^p \cdot \max_{i\in[n]} x_{ij}$. Similarly, to implement $\boldsymbol{x}$, good j should be duplicated with a minimum of $\sum_{i\in[n]} x_{ij}$ times in expectation, thus $c^d \cdot \sum_{i\in[n]} x_{ij}$ is the expected duplication cost for good j.

We refer $F^{c^p,c^d}_{n,m}$ as the problem with n participants, m goods, production cost c^p and duplication cost c^d for each good, and type distribution F. We select type distribution from $F \in \{U, B, C\}$, representing that the type distribution is i.i.d. from $U([0, 1])$, $Ber(p = 0.5)$, or correlated, respectively. For correlated distribution, we first generate $t_j \overset{\text{i.i.d.}}{\sim} U([0.25, 0.75])$ for each good j and then generate $t_{ij} \overset{\text{i.i.d.}}{\sim} U([t_j - 0.25, t_j + 0.25])$ for each participant i.

4.2 Baselines

4.2.1 Learning-Based Approaches. We choose the representative approach from the differentiable economics literature as baselines, respectively: RegretNet [18], Lottery-AMA [9] and GemNet [50]. These approaches were originally designated for unit-supply auctions, and we minimally modify these approaches to fit our setting and denote them as MD-RegretNet, MD-LotteryAMA, and MD-GemNet[8], respectively.

- MD-RegretNet [19]: Parameterize the allocation rule and payment rule directly with neural networks. The measure of untruthfulness is penalized during training.
- MD-LotteryAMA [9]: Learn an AMA mechanism with the discretization of global outcome space and boosting values.
- MD-GemNet [50]: Learn a discrete menu for each participant, where each menu is output by a neural network taking others' types as input.

For these baselines, we align the hyper-parameters with TEDI to make a fair comparison, including batch size, network iterations, temperature, *etc.* We note that when there is a single participant, GemNet and Lottery-AMA reduce to MenuNet [44] and RochetNet [19]. Therefore, we merge them as MD-MenuNet when $n = 1$.

4.2.2 Traditional Approaches. We consider three traditional baselines as follows.

- VCG [48]: The most classical mechanism with strong versatility.
- Bundle-OPT: Bundling all goods together at a parameterized reserve price, which is optimized through grid search. This baseline is also used in LotteryAMA [9].
- Separable-OPT: Drawing upon but different from a widely-used baseline, Item-wise Myerson [9, 18, 50]. Separable-OPT runs optimal mechanism on each good separately, where all optimal mechanisms for 1-good settings are derived by hands.

4.2.3 Ablation Studies. We conduct four ablation studies to show how each component in TEDI will affect its performance. The first operates on learning algorithm and the remaining operates on network architectures.

- TEDI-Discrete: Discretize the local outcome space as MD-GemNet does. The payments are output by Partial GroupMax Network, with local outcomes as inputs.
- TEDI-PICNN: Replace Partial GroupMax Network with PICNN (Partial Input Convex Neural Network) [2]. PICNN also parameterizes partial convex functions, but it has no universal approximation guarantee.
- TEDI-PMA: Replace Partial GroupMax Network with PMA (Parameterized Max-Affine) [29]. PMA parameterizes partial convex functions, but it has incomplete network architecture.
- TEDI-MLP: Replace Partial GroupMax Network with an MLP. MLP has no convexity guarantee on its inputs.

4.3 Experimental Results and Analysis

Table 1 presents all experimental results with up to $n = 10$ participants and $m = 30$ goods, respectively. The optimal values among all approaches under comparison are highlighted in bold. The optimal values among all truthful, learning-based approaches are underlined (this excludes MD-RegretNet from comparison). The superscript (*) represents the optimal value known for each setting.

4.3.1 Comparison between TEDI and Baselines. For these experiments, TEDI performs optimally 9 out of 12 times among all truthful approaches and 10 out of 12 times among all truthful, learning-based approaches. Figure 1 in Section 1 presents an illustration of TEDI and MD-MenuNet in the setting of $B_{1,10}$, showing that while TEDI successfully learns the optimal scheme of selling goods independently, MD-MenuNet fails. Another notable comparison is the settings of $U^{1,0}_{3,m}$, where MD-GemNet outperforms TEDI in $m = 2$ but TEDI outperforms MD-GemNet (as well as other algorithms) as m increases. These comparisons demonstrate that while existing algorithms suffer from curse-of-dimensionality, TEDI can result in strong performance when the problem size becomes large. Other experiments demonstrate that TEDI consistently results in good performance in diverse settings, with reasonable computational costs and varying problem contexts.

We also point out some (seemingly surprising) phenomena about baselines: 1) MD-LotteryAMA performs extremely worse in some settings ($U^{0,0.5}_{3,10}$, $C^{0,0.5}_{3,10}$) and even worse than Separable-OPT in the

[8] MD stands for "Mechanism Design".

Table 1: Experimental results with diverse approaches (including ablations studies) and diverse settings. '=0' ('<0') represents the value is exactly 0 (smaller than 0); '-' represents that the approach (row) does not apply to the setting (column). '*' represents the optimal value known for each setting. The *GPU Time* and *TFLOPS* results are measured when the model has $n = 3$ players and $m \in \{10, 30\}$ goods. The first (second) value represents the result for $m = 10$ ($m = 30$) respectively. All approaches have consistent hyper-parameters and random seeds in different settings except for MD-GemNet and MD-RegretNet, where hyper-parameters and random seeds are fine-tuned and the best-performed one is selected.

Methods\Settings	$U^{0,0}_{1,10}$	$B^{0,0}_{1,10}$	$U^{1,0}_{3,2}$	$U^{1,0}_{3,10}$	$U^{1,0}_{3,20}$	$U^{1,0}_{3,30}$	$U^{0,0.5}_{3,10}$	$U^{0,0.5}_{3,20}$	$U^{0,0.5}_{3,30}$	$B^{1,0}_{3,10}$	$C^{1,0}_{3,10}$	$C^{0,0.5}_{3,10}$	$U^{0,0.5}_{10,10}$	GPU Time (s)	TFLOPS
TEDI	3.4620	4.9128	0.2678	1.7285	3.8436	5.8771	**2.1989**	**4.6609**	**7.1884**	5.8090	2.2400	**1.4101**	**7.3140**	1,731 & 2,965	390 & 5,600
TEDI-Discrete	3.4292	3.4932	0.2463	1.4221	-	-	=0	-	-	4.2375	1.7460	=0	-	739 & -	103 & -
TEDI-PICNN	3.4527	4.7256	0.1288	0.9798	-	-	2.0337	-	-	5.7548	1.5281	1.2192	-	1,425 & -	340 & -
TEDI-PMA	2.4867	4.9619	0.2297	0.8976	-	-	1.8424	-	-	5.7047	1.0720	1.1570	-	2,211 & -	467 & -
TEDI-MLP	=0	=0	=0	=0	-	-	=0	-	-	0.2474	=0	=0	-	505 & -	170 & -
MD-MenuNet	3.4472	4.0565	-	-	-	-	-	-	-	-	-	-	-	-	-
MD-GemNet	-	-	0.2748	1.4814	2.5325	5.2015	1.2848	1.8463	2.1646	1.3309	1.7893	0.6350	4.6830	1,025 & 6,505	467 & 8,495
MD-LotteryAMA	-	-	0.2212	1.2050	2.5186	5.1290	0.3089	0.4914	0.3983	1.8118	1.7017	0.2286	< 0	708 & 2,837	106 & 2,013
MD-RegretNet	**3.5186**	4.9444	**0.4053**	**1.9053**	**5.0585**	**9.2974**	2.1900	2.9527	0.4824	5.6432	**2.9853**	1.0874	3.1285	8,793 & 9,023	552 & 9,043
(regret)	0.0031	0.0141	0.0507	0.0409	0.2038	0.3080	0.2852	0.3664	0.0761	0.0171	0.0756	0.1537	0.7762	-	-
VCG	=0	=0	<0	<0	<0	<0	=0	=0	=0	<0	<0	=0	= 0	-	-
Separable-OPT	2.5000	**5.0000***	0.2601	1.3019	2.6019	3.9059	1.8752	3.7501	5.6221	**6.2507**	1.4981	1.1894	6.2537	-	-
Bundle-OPT	3.4477	3.3059	=0	0.3616	2.5718	5.3308	0.4709	0.6652	0.8144	=0	1.4033	0.3319	1.5754	-	-

2-goods settings, $U^{1,0}_{3,2}$. This result discloses the limited expressiveness of affine maximizer mechanisms [37], which is consistent with existing results [7, 43]; 2) MD-GemNet (as well as TEDI-Discrete) has 0 value in some random seeds. We speculate that this phenomenon is a joint impact of following two factors: a) the naive, allocate-nothing mechanism is a saddle point (since we introduce the cost function), and b) discretization-based approaches are more sensitive to parameter disturbance [9, 17], thus is prone to stuck in saddle points. 3) MD-RegretNet has extremely large regret and smaller value in some settings (*e.g.*, $U^{0,0.5}_{3,10}, C^{0,0.5}_{3,10}$). Notably, in the $U^{0,0.5}_{3,30}$ setting, MD-RegretNet fails to learn a mechanism with both high revenue and low regret, despite our attempts at extensive hyperparameter tuning. We speculate that this happens because MD-RegretNet needs to compute regret in high-dimensional type space ($\mathcal{T}_i$). However, computing the maximum point *w.r.t.* a non-concave function in high-dimensional space is highly unstable [12, 25, §8].

4.3.2 Analysis of Ablation Studies. We conduct experiments with $n \le 3, m \le 10$ for ablation studies. We have following observations: 1) Comparing with TEDI-Discrete, TEDI generalizes the discrete outcome candidates to full spaces ($\mathcal{X}_i$). We note that TEDI outperforms TEDI-Discrete with a significant gap. This comparison suggests that replacing the discretization step plays an important role on improving the performance. 2) The performance gap between TEDI and TEDI-PICNN suggests that PICNN [2] is highly unlikely to be universal approximator of partial convex functions, although the existence of counter-examples are still open. 3) The performance of TEDI-PMA is close to and slightly worse than Separable-OPT. We speculate that this arises from following reasons: a) Since TEDI-PMA exhibits incomplete network architecture, it is likely to simply learn a linear pricing rule in high-dimensional space, and b) Separable-OPT always gives the optimal linear pricing rule for those settings presented in Table 1. 4) The extremely-poor performance of TEDI-MLP suggests that guaranteeing the partial convexity of pricing rule is of vital importance. We think the most important reason to be as follows: The fast convergence of Langevin dynamics requires the distribution $q_i^{\beta_i}(\cdot; \boldsymbol{t}_{-i}, \theta)$ be log-concave [42], and it gets satisfied only when pricing rule is partially convex.

5 CONCLUSIONS AND FUTURE WORKS

To conclude this paper, we propose TEDI, a discretization-free algorithm that can learn truthful mechanisms. By leveraging Partial GroupMax Network architecture as well as two techniques for deriving unbiased gradient estimators, our approach achieves truthfulness, full-expressiveness, and dimension-insensitivity, overcoming the "curse of dimensionality" that limits existing discretization-based methods and showing its competitive empirical performance in large-scale scenarios. We believe that the proposed network and learning paradigm contribute to the understanding of automated mechanism design and broader fields.

We note that TEDI can not be directly applied to learn feasible unit-supply auctions studied in [9, 19, 50] by nature of menu mechanisms. For future works, it is promising to extend the proposed techniques to learn large-scale unit-supply auctions by parameterizing the boosting function in AMA mechanisms [37] with universally convex neural networks, thus potentially improving the direct discretization as in prior works [9, 14]. Besides, when participants or goods are homogeneous, the problem has "permutation invariance property" [26, 31, 53]. Designing permutation-invariant pricing rule may likely accelerate the training process of TEDI and decrease the risk of over-fitting [13, 40]. For this reason, we hope that this study will motivate future explorations on the universal approximators of convex and permutation-invariant functions.

ACKNOWLEDGEMENTS

Yukun Cheng and Xiaotie Deng are the corresponding authors. This work is supported by the National Natural Science Foundation of China (Nos. 12471339, 62572010). Yunxuan Ma would like to thank Ruosong Wang, Lei Wu and Haoran Sun from Peking University for helpful discussions.

REFERENCES

[1] 2025. Reproducible and Non-Reproducible Goods - Okpedia. https://www.okpedia.com/reproducible-and-non-reproducible-goods

[2] Brandon Amos, Lei Xu, and J Zico Kolter. 2017. Input convex neural networks. In *International conference on machine learning*. PMLR, 146–155.

[3] Richard Bellman. 1966. Dynamic programming. *science* 153, 3731 (1966), 34–37.

[4] Yoshua Bengio, Olivier Delalleau, and Nicolas Roux. 2005. The curse of highly variable functions for local kernel machines. *Advances in neural information processing systems* 18 (2005).

[5] Léon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers*. Springer, 177–186.

[6] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

[7] Juan Carlos Carbajal, Andrew McLennan, and Rabee Tourky. 2013. Truthful implementation and preference aggregation in restricted domains. *Journal of Economic Theory* 148, 3 (2013), 1074–1101.

[8] Ning Chen, Nick Gravin, and Pinyan Lu. 2014. Optimal competitive auctions. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*. 253–262.

[9] Michael Curry, Tuomas Sandholm, and John Dickerson. 2023. Differentiable economics for randomized affine maximizer auctions. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. 2633–2641.

[10] Michael J Curry, Zhou Fan, and David C Parkes. 2024. Optimal Automated Market Makers: Differentiable Economics and Strong Duality. *arXiv preprint arXiv:2402.09129* (2024).

[11] Constantinos Daskalakis, Alan Deckelbaum, and Christos Tzamos. 2015. Strong duality for a multiple-good monopolist. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*. 449–450.

[12] Yann N Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. 2014. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *Advances in neural information processing systems* 27 (2014).

[13] Zhijian Duan, Yunxuan Ma, and Xiaotie Deng. 2023. Are equivariant equilibrium approximators beneficial?. In *International Conference on Machine Learning*. PMLR, 8747–8778.

[14] Zhijian Duan, Haoran Sun, Yurong Chen, and Xiaotie Deng. 2024. A scalable neural network for DSIC affine maximizer auction design. *Advances in Neural Information Processing Systems* 36 (2024).

[15] Zhijian Duan, Haoran Sun, Yichong Xia, Siqiang Wang, Zhilin Zhang, Chuan Yu, Jian Xu, Bo Zheng, and Xiaotie Deng. 2024. Automated Deterministic Auction Design with Objective Decomposition. *arXiv preprint arXiv:2402.11904* (2024).

[16] Paul Duetting, Vahab Mirrokni, Renato Paes Leme, Haifeng Xu, and Song Zuo. 2024. Mechanism design for large language models. In *Proceedings of the ACM on Web Conference 2024*. 144–155.

[17] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. 2015. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679* (2015).

[18] Paul Dütting, Zhe Feng, Harikrishna Narasimhan, David Parkes, and Sai Srivatsa Ravindranath. 2019. Optimal auctions through deep learning. In *International Conference on Machine Learning*. PMLR, 1706–1715.

[19] Paul Dütting, Zhe Feng, Harikrishna Narasimhan, David C Parkes, and Sai Srivatsa Ravindranath. 2024. Optimal auctions through deep learning: Advances in differentiable economics. *J. ACM* 71, 1 (2024), 1–53.

[20] Stuart Geman and Donald Geman. 1984. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence* 6 (1984), 721–741.

[21] Yiannis Giannakopoulos and Elias Koutsoupias. 2014. Duality and optimality of auctions for uniform distributions. In *Proceedings of the fifteenth ACM conference on Economics and computation*. 259–276.

[22] Andrew V Goldberg and Jason D Hartline. 2001. Competitive auctions for multiple digital goods. In *European Symposium on Algorithms*. Springer, 416–427.

[23] Andrew V Goldberg, Jason D Hartline, Anna R Karlin, and Michael Saks. 2004. A lower bound on the competitive ratio of truthful auctions. In *STACS 2004: 21st Annual Symposium on Theoretical Aspects of Computer Science, Montpellier, France, March 25-27, 2004. Proceedings 21*. Springer, 644–655.

[24] Noah Golowich, Harikrishna Narasimhan, and David C. Parkes. 2018. Deep learning for multi-facility location mechanism design *(IJCAI'18)*. AAAI Press, 7.

[25] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[26] Jiequn Han, Yingzhou Li, Lin Lin, Jianfeng Lu, Jiefu Zhang, and Linfeng Zhang. 2022. Universal approximation of symmetric and anti-symmetric functions. *Communications in Mathematical Sciences* 20, 5 (2022), 1397–1408.

[27] Zhiyi Huang and Anthony Kim. 2019. Welfare maximization with production costs: A primal dual approach. *Games and Economic Behavior* 118 (2019), 648–667.

[28] Herman Kahn and Theodore E Harris. 1951. Estimation of particle transmission by random sampling. *National Bureau of Standards applied mathematics series* 12 (1951), 27–30.

[29] Jinrae Kim and Youdan Kim. 2022. Parameterized convex universal approximators for decision-making problems. *IEEE Transactions on Neural Networks and Learning Systems* 35, 2 (2022), 2448–2459.

[30] Teun Kloek and Herman K Van Dijk. 1978. Bayesian estimates of equation system parameters: an application of integration by Monte Carlo. *Econometrica: Journal of the Econometric Society* (1978), 1–19.

[31] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*. PMLR, 3744–3753.

[32] Yunxuan Ma, Yide Bian, Hao Xu, Weitao Yang, Jingshu Zhao, Zhijian Duan, Feng Wang, and Xiaotie Deng. 2025. Large-Scale Contextual Market Equilibrium Computation through Deep Learning. In *International Workshop on Frontiers in Algorithmics*. Springer, 356–371.

[33] Yunxuan Ma, Siqiang Wang, Zhijian Duan, Yukun Cheng, and Xiaotie Deng. 2025. Learning Truthful Mechanisms without Discretization. *arXiv preprint arXiv:2506.22911* (2025).

[34] Alejandro M Manelli and Daniel R Vincent. 2006. Bundling as an optimal selling mechanism for a multiple-good monopolist. *Journal of Economic Theory* 127, 1 (2006), 1–35.

[35] Andreu Mas-Colell, Michael Dennis Whinston, Jerry R Green, et al. 1995. *Microeconomic theory*. Vol. 1. Oxford University Press New York.

[36] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics* 21, 6 (1953), 1087–1092.

[37] Paul R Milgrom and Robert J Weber. 1982. A theory of auctions and competitive bidding. *Econometrica: Journal of the Econometric Society* (1982), 1089–1122.

[38] Harikrishna Narasimhan, Shivani Brinda Agarwal, and David C Parkes. 2016. Automated mechanism design without money via machine learning. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*.

[39] Gregory Pavlov. 2011. Optimal mechanism for selling two goods. *The BE Journal of Theoretical Economics* 11, 1 (2011), 000010220193517041664.

[40] Tian Qin, Fengxiang He, Dingfeng Shi, Wenbing Huang, and Dacheng Tao. 2022. Benefits of permutation-equivariance in auction mechanisms. *Advances in Neural Information Processing Systems* 35 (2022), 18131–18142.

[41] Sai Srivatsa Ravindranath, Zhe Feng, Shira Li, Jonathan Ma, Scott D Kominers, and David C Parkes. 2021. Deep learning for two-sided matching. *arXiv preprint arXiv:2107.03427* (2021).

[42] Gareth O Roberts and Richard L Tweedie. 1996. Exponential convergence of Langevin distributions and their discrete approximations. (1996).

[43] Tuomas Sandholm and Anton Likhodedov. 2015. Automated Design of Revenue-Maximizing Combinatorial Auctions. *Operations Research* 63, 5 (2015), 1000–1025.

[44] Weiran Shen, Pingzhong Tang, and Song Zuo. 2019. Automated Mechanism Design via Neural Networks. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. 215–223.

[45] Ermis Soumalias, Michael J Curry, and Sven Seuken. 2024. Truthful Aggregation of LLMs with an Application to Online Advertising. *arXiv preprint arXiv:2405.05905* (2024).

[46] Haoran Sun, Yurong Chen, Siwei Wang, Wei Chen, and Xiaotie Deng. 2024. Mechanism Design for LLM Fine-tuning with Multiple Reward Models. *arXiv preprint arXiv:2405.16276* (2024).

[47] Haoran Sun, Xuanzhi Xia, Xu Chu, and Xiaotie Deng. 2025. Enhancing Affine Maximizer Auctions with Correlation-Aware Payment. (2025).

[48] William Vickrey. 1961. Counterspeculation, auctions, and competitive sealed tenders. *The Journal of finance* 16, 1 (1961), 8–37.

[49] Tonghan Wang, Paul Duetting, Dmitry Ivanov, Inbal Talgam-Cohen, and David C Parkes. 2024. Deep contract design via discontinuous networks. *Advances in Neural Information Processing Systems* 36 (2024).

[50] Tonghan Wang, Yanchen Jiang, and David C Parkes. 2024. GemNet: Menu-Based, Strategy-Proof Multi-Bidder Auctions Through Deep Learning. In *Proceedings of the 25th ACM Conference on Economics and Computation*. 1100.

[51] Xavier Warin. 2023. The GroupMax neural network approximation of convex functions. *IEEE Transactions on Neural Networks and Learning Systems* (2023).

[52] Andrew Chi-Chih Yao. 2017. Dominant-strategy versus bayesian multi-item auctions: Maximum revenue determination and comparison. In *Proceedings of the 2017 ACM Conference on Economics and Computation*. 3–20.

[53] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. 2017. Deep sets. *Advances in neural information processing systems* 30 (2017).